REMARKS

Reconsideration of the application as amended is respectfully requested. The enclosed is responsive to the Examiner's Office Action mailed October 20, 2010. By way of the present response, applicants have 1) amended claims 1, 9, 15, and 19; 2) canceled claims 3, 11 and 18; and 3) added no new claims. Claims 22-25 were been previously withdrawn in a preliminary amendment mailed on September 28, 2006. The specification has been amended to correct typographical errors. No new matter has been added.

Applicants reserve all rights with respect to the applicability of the Doctrine of Equivalents

Claim Objections

The Examiner has objected to claim 1 because alleged informalities. The Examiner has stated that the phrase "beginning initialization a foreign [sic] thread" in line 2 is grammatically incorrect. Applicants respectfully submit that claim 1, line 2 has been amended to recite "beginning initialization of a first thread. . .," and is now grammatically correct.

The Examiner has objected to claims 1 and 19 because the term "capable of" allegedly invokes the intended use, which does not positively recite the claims. Applicants respectfully submit that claims 1 and 19 have been amended to positively recite those claim features.

Claim Rejections – 35 U.S.C. § 101

Claims 15-18 have been rejected under 35 U.S.C. § 101 as being directed to non-statutory subject matter. Applicants respectfully submit that claim 15 has been amended to fall under one of the four statutory categories of invention under 35 U.S.C. § 101. Consequently, claims 16-18, which depend from claim 15, also fall under one of the four statutory categories of invention under 35 U.S.C. § 101.

Claim Rejections – 35 U.S.C. § 102(e)

Claims 1-5 stand rejected under U.S.C. §102(e) as being anticipated by U.S. Patent Pub. No. 2005/0120194 of Kissell ("Kissell"). Applicants reserve the right to swear behind Kissell at a later date. Applicants respectfully request withdrawal of these rejections because the cited reference fails to teach or suggest all of the features of the claims.

With respect to claim 1, Kissell fails to disclose:

A computer implemented method comprising:

beginning initialization of a first thread from a second context, wherein the first thread executable on a first context and the second context;

suspending the initialization of the first thread at a position within the second context, wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread;

creating a second thread based on the position in the second context; and

completing the initialization of the first thread continuing from the position in the second context.

(Emphasis added).

Applicants respectfully submit that Kissell fails to disclose at least "suspending the initialization of the first thread at a position within the second context, wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread," as recited in amended claim 1.

Kissell describes a fork instruction for execution on a multithreaded microprocessor while occupying a single instruction issue slot. (Kissell, Abstract). Kissell further describes multithreaded processors including multiple sets of resources, or contexts, for storing unique states of each thread, thereby facilitating the ability to quickly switch between threads to fetch and issue instructions. (Kissell, paragraph 7). The Examiner asserts that switching between contexts in a multithreaded microprocessor is the same as suspending the initialization of the first thread at a position within the second context, wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread. (Office Action, 10/20/10, pg. 3-4, *see* comments for claims 1 and 3). Applicants respectfully disagree with this characterization of the cited reference. Although Kissell describes switching between threads to fetch and issue instructions and interrupts for service requests, Kissell makes no mention of an "interrupt of the context [resource] switching," as proffered by the Examiner. (*Id.*). If the Examiner meant to say that "interrupt of the *thread* switching," as described by Kissell anticipates features of claim 1, applicants respectfully disagree with this characterization as well. Threads are independent from one another in a multithreaded processor environment, thus allowing concurrent processing. (Kissell, paragraph 9). As such, no such suspension occurs.

For the sake of argument, even if Kissell did describe an interrupt due to context switching, the fact that multiple contexts are assigned per thread implies that the context assignment has already occurred. (Kissell, paragraph 7). In other words, in order to switch between multiple contexts of a particular thread, the contexts (resources) must have already been assigned to that particular thread in order to facilitate switching between them. Put simply, the initialization (thread creation and resource assignment) of a particular thread must be complete before switching between thread contexts (resources) for the execution of that thread can occur. As such, Kissell fails to disclose "suspending the initialization of the first thread at a position within the second context," as recited in claim 1.

Furthermore, although Kissell describes both switching between threads to fetch and issue instructions (*Id.*) and allocating context for a new thread in an instruction which occupies a single instruction issue slot (Kissell, paragraph 11), Kissell fails to disclose "wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread," as recited in claim 1.

Accordingly, applicants respectfully submit that the rejection of claim 1 under 35 U.S.C. § 102(e) has been overcome and respectfully requests the withdrawal of the rejection of the claim.

Given the claims 2 and 4 depend upon claim 1, and include additional features, applicants respectfully submit that the rejection of claims 2 and 4 under 35 U.S.C. § 102(e) has been overcome for at least the same reasons and respectfully requests the withdrawal of the rejection of the claims.

Claims 15-19 stand rejected under U.S.C. §102(e) as being anticipated by U.S. Patent Pub. No. 2006/0184920 to Wang ("Wang"). Applicants reserve the right to swear behind Wang at a later date. Applicants respectfully request withdrawal of these rejections because the cited reference fails to teach or suggest all of the features of the claims.

With respect to amended claim 15, Wang fails to disclose:

A computer system for managing thread resources comprising:
a first multithreaded programming environment, executed by a processor;
a second multithreaded programming environment, executed by the processor;
a multithreaded program, executed by the processor, including a first thread and a second thread;
a host platform; and
a dynamic binary translator to translate the first thread for the first

multithreaded programming environment to be executed in the second multithreaded programming environment, wherein **the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected.**

(Emphasis added).

Applicants respectfully submit that Wang fails to disclose at least "the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected," as recited in amended claim 15.

Wang describes supporting the execution of a managed application that is linked to a native library or application. (Wang, Abstract) Wang further describes support for a virtual machine that is associated with the same instruction set architecture (ISA) as the executing platform, while the ISA of the native library or application is of a different ISA. (*Id.*). Although Wang describes intercepting and translating calls to reference objects, such as converting 32-bit calls into 64-bit calls, Wang is silent on suspending operating system requests from thread when requests to create a new thread are detected. As such, Wang fails to disclose "the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected," as recited in claim 15.

Accordingly, applicants respectfully submit that the rejection of claim 15 under 35 U.S.C. § 102(e) has been overcome and respectfully requests the withdrawal of the rejection of the claim.

Given the claims 16-17 depend upon claim 15, and include additional features, applicants respectfully submit that the rejection of claims 16-17 under 35 U.S.C. § 102(e) has been overcome for at least the same reasons and respectfully requests the withdrawal of the rejection of the claims.

With respect to amended claim 19, Wang fails to disclose:

A computer system for managing thread resource comprising:
a random accessed memory;
a first processor configured to execute multithreaded programs stored in the random accessed memory;
a multithreaded program to be executed on a second processor; and

a program to transparently initialize and create a thread included in the multithreaded program in an environment supported by the first processor to be executed on the second processor, **wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread.**

(Emphasis added).

Applicants respectfully submit that Wang fails to disclose at least "wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread," as recited in amended claim 19.

As described above, Wang describes intercepting and translating calls to reference objects, such as converting 32-bit calls into 64-bit calls on multithreading microprocessors. Nevertheless, Wang does not disclose "wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread," as recited in claim 19.

Accordingly, applicants respectfully submit that the rejection of claim 19 under 35 U.S.C. § 102(e) has been overcome and respectfully requests the withdrawal of the rejection of the claim.

Claim Rejections – 35 U.S.C. § 103

Claims 6-18 stand rejected under U.S.C. §103(a) as being unpatentable over Kissell in view of "64-bit versus 32-bit Virtual Machines for Java," to Venstermans et al. ("Venstermans"). Applicants reserve the right to swear behind Venstermans at a later date. Applicants respectfully request withdrawal of these rejections.

Given that claims 6-8 depend from claim 1, and include additional features, applicants respectfully submit that the rejection of claims 6-8 under 35 U.S.C. § 103(a) has been overcome for at least the same reasons as described above and respectfully requests the withdrawal of the rejection of the claims.

With respect to amended claim 9, the combination of cited references fails to disclose:

A computer implemented method comprising:

beginning initialization of a foreign thread from a host platform, wherein the foreign thread is to be executed on a foreign platform;

suspending the initialization of the foreign thread at a position within the host platform, wherein the beginning initialization of the foreign thread is

suspended in response to a detection of an operating system request to create the foreign thread;

recording the position of the suspension;

creating a host thread from a host platform based on the recorded position; and

completing the initialization of the foreign thread continuing from the recorded position in the host platform.

(Emphasis added).

Applicants respectfully submit that the combination of Kissell and Venstermans fails to teach or suggest "suspending the initialization of the foreign thread at a position within the host platform, wherein the beginning initialization of the foreign thread is suspended in response to a detection of an operating system request to create the foreign thread," as recited in claim 9.

Kissell describes thread switching to fetch and issue instructions and interrupts for service requests, not "suspending the initialization of [a] foreign thread at a position within [a] host platform, wherein the beginning initialization of the foreign thread is suspended in response to a detection of an operating system request to create the foreign thread," as recited in claim 9.

Venstermans fails to remedy the shortcomings of Kissell with respect to teaching or suggesting all of the features of claim 9. Venstermans teaches platform independent properties of Java to allow, for example, the same Java bytecode to run properly on 32-bit or 64-bit virtual machines (VM's). (Venstermans, Summary). Vensterman does not, however, teach or suggest "suspending the initialization of [a] foreign thread at a position within [a] host platform, wherein the beginning initialization of the foreign thread is suspended in response to a detection of an operating system request to create the foreign thread," as recited in claim 9.

Accordingly, applicants respectfully submit that the rejection of claim 9 under 35 U.S.C. § 103(a) has been overcome and respectfully requests the withdrawal of the rejection of the claim.

Given the claims 10 and 12-14 depend upon claim 9, and include additional features, applicants respectfully submit that the rejection of claims 10 and 12-14 under 35 U.S.C. § 103(a) has been overcome for at least the same reasons and respectfully requests the withdrawal of the rejection of the claims.

With respect to amended claim 15, the combination of cited references fails to disclose:

A computer system for managing thread resources comprising:

a first multithreaded programming environment, executed by a processor;

a second multithreaded programming environment, executed by the processor;

a multithreaded program, executed by the processor, including a first thread and a second thread;

a host platform; and

a dynamic binary translator to translate the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment, **wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected.**

(Emphasis added).

Applicants respectfully submit that the combination of Kissell and Venstermans fails to teach or suggest "wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected," as recited in claim 15.

As described above, Kissell describes thread switching to fetch and issue instructions and interrupts for service requests. (Kissell, paragraph 7). Kissell does not teach or suggest "wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected," as recited in claim 15.

Venstermans fails to remedy the shortcomings of Kissell with respect to teaching or suggesting all of the features of claim 15. As described above, Venstermans teaches platform independent properties of Java to allow, for example, the same Java bytecode to run properly on 32-bit or 64-bit virtual machines (VM's). (Venstermans, Summary). Vensterman does not, however, teach or suggest "wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected," as recited in claim 15.

Accordingly, applicants respectfully submit that the rejection of claim 15 under 35 U.S.C. § 103(a) has been overcome and respectfully request the withdrawal of the rejection of the claim.

Given the claims 16-17 depend directly or indirectly upon claim 15, and include additional features, applicants respectfully submit that the rejection of claims 16-17 under 35 U.S.C. § 103(a) has been overcome for at least the same reasons and respectfully requests the withdrawal of the rejection of the claims.

Claims 19-21 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Kissell. Applicants respectfully request withdrawal of these rejections.

With respect to claim 19, applicants respectfully submit that Kissell fails to teach or suggest "wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread," as recited in claim 19.

As described above, Kissell describes thread switching to fetch and issue instructions and interrupts for service requests. (Kissell, paragraph 7). Kissell does not teach or suggest "wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread," as recited in claim 19.

Accordingly, applicants respectfully submit that the rejection of claim 19 under 35 U.S.C. § 103(a) has been overcome and respectfully requests the withdrawal of the rejection of the claim.

Given the claims 20-21 depend directly or indirectly upon claim 19, and include additional features, applicants respectfully submit that the rejection of claims 20-21 under 35 U.S.C. § 103(a) has been overcome for at least the same reasons and respectfully requests the withdrawal of the rejection of the claims.

Claims 1-21 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Kissell in view of "Java JNI Bridge: A Framework for Mixed native ISA Execution," to Chen et al. ("Chen"). Applicants respectfully request withdrawal of these rejections. Applicants reserve the right to swear behind Chen at a later time.

As described above with respect to claim 1, Kissell describes thread switching to fetch and issue instructions and interrupts for service requests. (Kissell, paragraph 7). Kissel does not disclose "suspending the initialization of the first thread at a position within the second context, wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread," as recited in claim 1.

Chen fails to remedy the shortcomings of Kissell with respect to teaching or suggesting all of the features of claim 1. Applicants agree with the Examiner that Chen does not expressly disclose "suspending the initialization of the first thread at a position within the second context," as recited in claim 1. (Office Action, 10/20/10, page 13).

Accordingly, applicants respectfully submit that the rejection of claim 1 under 35 U.S.C. § 103(a), with respect to Chen, has been overcome and respectfully requests the withdrawal of the rejection of the claim.

Given the claims 2 and 4-8 depend directly or indirectly upon claim 1, and include additional features, applicants respectfully submit that the rejection of claims 2 and 4-8 under 35 U.S.C. § 103(a) has been overcome for at least the same reasons and respectfully requests the withdrawal of the rejection of the claims.

With respect to claim 9, the Office Action rejected the claim for the same reasons as stated in the rejection of claim 1.

As described above with respect to claim 1, Kissell describes thread switching to fetch and issue instructions and interrupts for service requests. Kissel does not disclose "suspending the initialization of [a] foreign thread at a position within [a] host platform, wherein the beginning initialization of the foreign thread is suspended in response to a detection of an operating system request to create the foreign thread," as recited in claim 9.

Chen fails to remedy the shortcomings of Kissell with respect to teaching or suggesting all of the features of claim 9. Chen describes transparently running Java applications containing native method calls to one ISA on a different ISA's Java platform. (Chen, Abstract). Chen further describes operating within the same operating system process that executes the application but with the support of a dynamic translator. (*Id.*). Chen does not, however, disclose "suspending the initialization of [a] foreign thread at a position within [a] host platform, wherein the beginning initialization of the foreign thread is suspended in response to a detection of an operating system request to create the foreign thread," as recited in claim 9.

Accordingly, applicants respectfully submit that the rejection of claim 9 under 35 U.S.C. § 103(a), with respect to Kissel in view of Chen, has been overcome and respectfully requests the withdrawal of the rejection of the claim.

Given the claims 10 and 12-14 depend directly or indirectly upon claim 9, and include additional features, applicants respectfully submit that the rejection of claims 10 and 12-14 under 35 U.S.C. § 103(a) has been overcome for at least the same reasons and respectfully requests the withdrawal of the rejection of the claims.

With respect to claim 15, applicants respectfully submit that the combination of Kissell and Chen fails to teach or suggest "wherein the binary translator further includes an operating

system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected," as recited in claim 15.

As described above, Kissell describes thread switching to fetch and issue instructions and interrupts for service requests. (Kissell, paragraph 7). Kissell does not teach or suggest "wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected," as recited in claim 15.

Chen fails to remedy the shortcomings of Kissell with respect to teaching or suggesting all of the features of claim 15. As described above, Chen describes transparently running Java applications containing native method calls to one ISA on a different ISA's Java platform and operating within the same operating system process that executes the application but with the support of a dynamic translator. (Chen, Abstract). Chen does not, however, disclose "wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected," as recited in claim 15.

Accordingly, applicants respectfully submit that the rejection of claim 15 under 35 U.S.C. § 103(a) has been overcome and respectfully request the withdrawal of the rejection of the claim.

Given the claims 16-17 depend directly or indirectly upon claim 15, and include additional features, applicants respectfully submit that the rejection of claims 16-17 under 35 U.S.C. § 103(a) has been overcome for at least the same reasons and respectfully requests the withdrawal of the rejection of the claims.

With respect to claim 19, applicants respectfully submit that the combination of Kissell and Chen fails to teach or suggest "wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread."

As described above, Kissell describes thread switching to fetch and issue instructions and interrupts for service requests. (Kissell, paragraph 7). Kissell does not teach or suggest "wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread," as recited in claim 19.

Chen fails to remedy the shortcomings of Kissell with respect to teaching or suggesting all of the features of claim 19. As described above, Chen describes transparently running Java

applications containing native method calls to one ISA on a different ISA's Java platform and operating within the same operating system process that executes the application but with the support of a dynamic translator. (Chen, Abstract). Chen does not, however, disclose "wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread," as recited in claim 19.

Accordingly, applicants respectfully submit that the rejection of claim 19 under 35 U.S.C. § 103(a) has been overcome and respectfully request the withdrawal of the rejection of the claim.

Given the claims 20-21 depend directly or indirectly upon claim 19, and include additional features, applicants respectfully submit that the rejection of claims 20-21 under 35 U.S.C. § 103(a) has been overcome for at least the same reasons and respectfully requests the withdrawal of the rejection of the claims.

CONCLUSION

It is respectfully submitted that in view of the amendments and arguments set forth herein, the applicable rejections and objections have been overcome.

If there are any additional charges, please charge Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: January 20, 2011

Lester J. Vincent
Reg. No. 31,460

Customer No. 08791

1279 Oakmead Parkway
Sunnyvale, CA 94085-4040
(408) 720-8300